NASA
Technical Memorandum 106750

Army Research Laboratory
Technical Report ARL–TR–598

An Object-Oriented Graphical User Interface for a Reusable Rocket Engine Intelligent Control System

Jonathan S. Litt
Vehicle Propulsion Directorate
U.S. Army Research Laboratory
Lewis Research Center
Cleveland, Ohio

Jeffrey L. Musgrave, Ten-Huei Guo, Daniel E. Paxson,
Edmond Wong, Joseph R. Saus, and Walter C. Merrill
Lewis Research Center
Cleveland, Ohio

October 1994

National Aeronautics and
Space Administration

ORIGINAL CONTAINS
COLOR ILLUSTRATIONS

AN OBJECT-ORIENTED GRAPHICAL USER INTERFACE FOR A REUSABLE ROCKET ENGINE INTELLIGENT CONTROL SYSTEM

Jonathan S. Litt
Vehicle Propulsion Directorate
U.S. Army Research Laboratory
Lewis Research Center
Cleveland, Ohio 44135

Jeffrey L. Musgrave, Ten-Huei Guo, Daniel E. Paxson, Edmond Wong,
Joseph R. Saus, and Walter C. Merrill
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

An Intelligent Control System for Reusable Rocket Engines under development at NASA Lewis Research Center requires a graphical user interface to allow observation of the closed-loop system in operation. The simulation testbed consists of a real-time engine simulation computer, a controls computer, and several auxiliary computers for diagnostics and coordination. The system is set up so that the simulation computer could be replaced by the real engine and the change would be transparent to the control system. Because of the hard real-time requirement of the control computer, putting a graphical user interface on it was not an option. Thus, a separate computer used strictly for the graphical user interface was warranted. An object-oriented LISP-based graphical user interface has been developed on a Texas Instruments Explorer II+ to indicate the condition of the engine to the observer through plots, animation, interactive graphics, and text.

NOMENCLATURE

ENGINE COMPONENTS

CCV	Chamber Coolant Valve
FPOV	Fuel Preburner Oxidizer Valve
HPFTP	High Pressure Fuel Turbopump
HPOTP	High Pressure Oxidizer Turbopump
LPFTP	Low Pressure Fuel Turbopump
MFV	Main Fuel Valve
MOV	Main Oxidizer Valve
OPFV	Oxidizer Preburner Fuel Valve
OPOV	Oxidizer Preburner Oxidizer Valve
SSME	Space Shuttle Main Engine

ENGINE VARIABLES

σ_{nom}	Nominal Stress
C_{tip}	Tip Clearance
D	Damage
$\dot{D}$	Damage Rate
DW_{ft2d}	High Pressure Fuel Turbine Fuel Flow
DW_{opl}	Low Pressure Oxidizer Pump Oxidizer Flow
MR	Mixture Ratio
P_5	Main Combustion Chamber Coolant Pressure
P_C	Chamber Pressure
P_{fdl}	Low Pressure Fuel Pump Outlet Pressure
P_{fp}	Fuel Preburner Pressure
P_{fs}	Fuel Supply Pressure
P_{ftld}	Low Pressure Fuel Turbine Discharge Pressure
P_{ft2d}	High Pressure Fuel Turbine Outlet Pressure
P_{ft2i}	High Pressure Fuel Turbine Inlet Pressure
P_{He}	Helium Purge Pressure
P_{op}	Oxidizer Preburner Pressure
P_{osd}	Oxidizer Seal Drain Pressure
P_{ot2d}	High Pressure Oxidizer Turbine Outlet Pressure
P_{psd}	Primary Seal Drain Pressure
P_{ssd}	Secondary Seal Drain Pressure
Q_{fdl}	Fuel Flow
S_{fl}	Low Pressure Fuel Turbopump Speed
S_{f2}	High Pressure Fuel Turbopump Speed
S_{o2}	High Pressure Oxidizer Turbopump Speed
T_5	Main Combustion Chamber Coolant Temperature
T_{fdl}	Low Pressure Fuel Pump Outlet Temperature
T_{ft2d}	High Pressure Fuel Turbine Outlet Temperature
T_{op}	Oxidizer Preburner Temperature
T_{osd}	Oxidizer Seal Drain Temperature
T_{ot2d}	High Pressure Oxidizer Turbine Outlet Temperature
T_{psd}	Primary Seal Drain Temperature

INTRODUCTION

The Reusable Rocket Engine Intelligent Control System (ICS) under development is generic technology which is being demonstrated on a model of the Space Shuttle Main Engine (SSME) [1], an example of a reusable rocket engine. The ICS testbed [2] consists of five interconnected computers as shown in figure 1. A real-time AD100 simulation computer runs a model of the engine [3] and associated valves and sensors, as well as models of the engine's behavior with failed components [4]. The Control Interface and Monitoring (CIM) Unit [5] is a control computer built in-house utilizing Intel 80486 technology. Several applications run on it including a reconfigurable controller and an engine level coordinator, both described in [6], and a model-based fault detection algorithm [7]. A VAX Station 3500 computer runs G2™ [8], a real-time

expert system shell developed for process monitoring and used here for diagnostics and rule-based fault detection [9]. A personal computer-mounted neural network processor board executes a sensor failure detection, isolation and accommodation scheme [10]. Finally, a Texas Instruments Explorer II+ runs the object-oriented graphical user interface (GUI).

All pertinent information and results such as sensed engine variable values and actuator, component, and sensor failures identified using the other computers are passed to the CIM Unit where they are used to compute the next control signal. The hard real-time constraint on the control computer makes it an unsuitable location for a time-consuming interactive graphical user interface. Therefore, an alternate platform was needed on which to run the GUI and a Texas Instruments Explorer II+ was selected. The Texas Instruments Explorer II+ LISP Machine has a large (16"), high resolution (1024 pixels × 754 pixels) color screen (up to 256 colors displayed simultaneously), an object-oriented window-based graphical system, a three-button mouse, and a sophisticated development environment including many graphics primitives, interactive debugging, a powerful screen editor, incremental compilation, and a LISP interpreter for interactive function evaluation. These features combined with the lack of a hard real-time requirement on the GUI make the Explorer an excellent platform for this application [11]. The Explorer system is shown in figure 2. The CIM Unit, which contains all known information relevant to the condition of the engine, is responsible for providing a snapshot of the current system status to the GUI via an ethernet connection so it can be displayed.

AN OBJECT-ORIENTED GRAPHICAL USER INTERFACE

An *object-oriented system* is one consisting of entities possessing certain data and operations [12]. These entities or *objects* interact in predefined ways to give the overall system the desired qualities. By object-oriented graphics, one means a set of graphical entities which have certain properties. These properties might include position, color, and size, for instance. Items traditionally thought of as graphical objects—polygons, sprites, blinkers—are only a fraction of the total. Other graphical objects used in this GUI include windows, frames, and the mouse cursor. Specific instances of a class of objects are created from a template called a *flavor* which is a generic object of a particular type with certain default properties. The new object will automatically inherit those properties unless they are explicitly set to something else. New flavors can be built by combining several existing types and the new ones will contain the properties of their parents. Objects communicate by sending *messages* to each other which, when received, produce a certain action. For example, a sprite is a graphical object which can move along a desired path on the screen automatically, i.e., the tasks of saving the background, drawing the sprite, erasing the sprite, redrawing the background, saving the background in the new location, redrawing the sprite in its new location, and so on, are done by the processor without instruction from the GUI. These actions can be initiated and altered by sending a message to the sprite, which is accomplished by executing a special line of code. The sprite object itself contains the program to update the screen and has it simply by virtue of being a sprite—it is an inherited property of all sprites.

COMMUNICATION WITH THE CIM UNIT

Twenty times a second, the CIM Unit sends a packet of data to the Explorer via an ethernet connection using the TCP/IP protocol. The Explorer has an area of memory called an *input buffer* which holds the data packets in a queue until they can be read sequentially by the GUI. The CIM Unit includes in each data packet a time stamp, 44 variable values to be plotted, and an integer indicating the failure status of each piece of hardware. The time stamp is used as the independent value in the coordinate axes against which the 44 variable values are plotted on the GUI screens described below. Some of the 44 variables are plotted on more than one screen. The binary integer containing the failure flags indicates to the GUI which faults have been identified. There are 19 identifiable failures, each indicated by a single bit, between 2^0 and 2^{18} (see Appendix). The integer containing the flags is sent in each data packet, even if it is unchanged, and the GUI compares it to the previous one it received. Any change indicates a new failure was detected. This makes the system robust since a packet on the network might be missed due to the input buffer overflowing and, if the flag were sent only once upon detection, it might be lost.

THE SCREENS

There are eight screens which make up the ICS GUI. Each screen consists of three windows built into a structure called a *frame*. Since there exists a one-to-one relationship between screens and frames in the ICS GUI, the words will be used interchangeably. The frames are arranged in a hierarchical, tree-like structure. The more general represent the base of the tree while the more specific represent its branches. Each frame contains at least one *icon*, a graphical object which symbolizes an action to be performed. In this case, clicking on the icon with the mouse exposes the frame corresponding to that icon. The hierarchy of screens and the paths of movement between them is shown in figure 3. The screens are described below.

The Mouse-Sensitive Graphics Window

The upper window on each screen (see figure 4, for example) is *mouse-sensitive*. This means that there are objects in the window which become highlighted when the mouse cursor is on them and that some action is performed when they are clicked on. In the ICS GUI, clicking on the selectable icon will bring up the screen which corresponds to it. When the mouse cursor is placed over a selectable object, a box appears around the object. Additionally, a text string indicating which frame will be exposed if the object is clicked on appears in the extreme lower left of the screen. Both the box and the text string disappear when the cursor is moved off the object. The mouse-sensitive graphics window is the one which contains the picture of the system or component so it is clear which icons might be selectable. To make it even clearer, whenever a failure occurs, the picture of the malfunctioning component begins blinking. The integer failure flag sent by the CIM Unit as part of the data packet initiates this blinking in the mouse-sensitive graphics windows. With this type of screen it is natural to select particular objects creating a smooth flow through the GUI to observe the engine's operation interactively.

The Plotting Window

Another graphics window is located at the lower right of each screen (see figure 4, for example). These plotting windows are not mouse-sensitive. They are animated to display the values of the

engine variables in strip-chart form updated in simulation time. Each of these windows contains from five to nine sets of coordinate axes displaying preselected variables appropriate to the picture in the mouse-sensitive graphics window on the same screen. Each axis can display up to 361 data points. When the plot comes to the right edge of the time-axis, it shifts left half way (the point at the right edge moves to the middle of the time-axis) and continues plotting to the right. Time is displayed across the top of the window and is updated with each leftward shift. Variable name, units, and range are included by each set of coordinate axes.

The plotting window assumes a user-specified number of 20 samples per second will be transferred from the CIM Unit and sets itself up so that the time-axis corresponds to 361 of those time steps (the time-axis is 361 pixels long). Ideally, the time stamps associated with the data sets correspond one-to-one to the pixels of the time-axis. If the GUI does not receive variable values for each expected time step, either because samples are sent at a slower rate or because some data packets get lost, the GUI will plot the points at their appropriate location based on the time stamp and linearly interpolate between the previous and current data point in the plotting window so spaces are not left blank. If the GUI receives samples at a faster rate than anticipated, implying that more than 361 data sets are received in the 18 seconds allotted, the array which saves the variable values will fill up prematurely causing a fatal error.

The Output Window

The output window is a LISP Listener, an interactive text window located in the lower left corner of the screen (see figure 4, for instance). System bulletins such as announcements of failures are broadcast to the output windows on all of the screens for informational purposes. The integer failure flag sent by the CIM Unit as part of the data packet initiates the broadcast of these descriptive bulletins. The output window accepts keyboard input, evaluates it and returns a value. Thus it can be used to examine or change variables within the GUI.

FAILURE DISPLAY

For each failure, a set of operations is performed. A bulletin is sent to the output window of each frame so there is some text visible to the user describing the fault. Messages are sent to all appropriate blinkers, telling them to flash. If a failure message appears in the output window of the screen being displayed but nothing is flashing in the mouse-sensitive graphics window, the user can return to the top level screen to see the blinker. From there, the user can proceed down through the tree of frames to the screen displaying the failure.

GUI SCREENS

The eight frames which make up the GUI and their functions are described below.

SSME Screen

The top level screen (figure 4) shows all major engine components, valves, and sensors. The oxidizer (oxygen) flow paths are shown in green and the fuel (hydrogen) flow paths are shown in red. Every failure which the ICS is able to account for can be indicated from this screen with a flasher in the upper window as well as the bulletin in the output window. Clicking on any of the valve icons exposes the Valve screen while clicking on any of the sensor name icons exposes

the Sensor screen. The LPFTP, HPOTP, and HPFTP Tip Seals screens can each be exposed by clicking on that component's icon. The data traces displayed are of the control-related variables of the SSME.

Valve Screen

The valve screen (figure 5) shows the same components as the SSME screen but the valves are enlarged and animated. The valves rotate in simulation time to show the percentage each is open. The plotted data also show the normalized valve position. All mouse-selectable items are the same as in the SSME screen (figure 4). Valve faults are recognized by the model-based fault detection system.

LPFTP Screen

The Low Pressure Fuel Turbopump screen (figure 6) shows a cross section of the turbopump. A failure is signaled on this screen when hydrogen gas leakage from the turbine to the pump is detected. This leakage causes a reduction in efficiency of the turbine. The rule-based fault detection scheme identifies this leakage using the variables displayed in the plotting window.

Sensor Screen

The Sensor screen (figure 7) depicts the sensor failure detection, isolation, and accommodation (FDIA) system. The FDIA system consists of nine sensors measuring variables on the fuel side of the engine. These nine variables are dependent upon each other, i.e. their values are interrelated such that any incorrect measurement can be detected given the other measurements and any missing measurement can be estimated given the others. In the ICS testbed, a neural network is used for this failure detection and data recovery. The upper left window on the sensor screen shows the fuel side of the engine with the sensor lines running into the input layer of the neural network. When a sensor failure flag is received from the CIM Unit, the sensor line turns red and is switched out and the previous output of the neural network corresponding to that sensor is switched in. Thus the GUI shows symbolically how the measurement is replaced by the estimate. Meanwhile, the plotting window displays the sensed value in black and the estimated value in red on the same graph.

HPFTP Tip Seal Screen

The High Pressure Fuel Turbopump tip seal screen (figure 8) shows the spinning turbine and a representation of the gap between the blade tips and the outer edge of the passage, which is an indicator of turbine efficiency. As the gap increases, more hot gas can leak through, bypassing the turbine blades thereby decreasing the useful energy. This is detected by the rule-based fault detection system using the efficiency-related variables plotted on the screen. The illusion of spinning is created by sprites moving along the turbine blades. In addition to displaying tip clearance, this screen contains the rotor blade icon used to expose the following screen.

HPFTP (Blade) Screen

The High Pressure Fuel Turbopump screen (figure 9) shows a first stage rotor blade, a detail of a section of the previous screen. The screen is used to give an indication of estimated remaining blade life. Exposure to repeated stress-strain cycling will cause fatigue cracks which will

eventually result in the blade severing [13]. The screen symbolizes this phenomenon by showing a crack propagating along the blade root until the instant of failure when the blade disappears leaving behind a jagged stump. The life estimation is performed by two neural networks in combination with numeric computations as shown symbolically in the figure, using the variables in the plotting window. It is not currently incorporated in the ICS test bed.

HPOTP Screen

The High Pressure Oxidizer Turbopump screen (figure 10) shows the cross section of the pump. This is an intermediate screen displaying HPOTP operating-condition variables and containing an icon which indicates the position of the seals in the following screen. No failures are associated uniquely with this screen.

HPOTP Seals Screen

The High Pressure Oxidizer Turbopump seals screen (figure 11) shows a close-up of the seal system. It is a detail of a section of the previous screen. A seal failure here could be disastrous as fuel-rich hot gas may come in contact with liquid oxygen resulting in an explosion. A failure is signaled as soon as the rule-based fault detection scheme, using the displayed variables, detects a broken seal.

OPERATIONAL MODES

The GUI is capable of operating in networked or stand-alone mode. The system was designed and is intended to run networked as part of the ICS testbed. However, during the development, testing and debugging phases, it is critical that the GUI have the ability to run on its own, duplicating its networked behavior. The user chooses the mode at start-up by calling the GUI routine with a logical argument indicating whether or not the system is networked. If the GUI is networked, the data values are read directly from the input buffer, if not, they are set within the program to some predetermined values. The two different data acquisition functions are the only code which is not common to both modes. The stand-alone mode allows a user to check every aspect of the GUI except for the reading of the input buffer. This way, all changes can be tested and evaluated without tying up the rest of the testbed system.

An automatic reset option was built in to avoid having to stop and restart the GUI and CIM Unit each time the control is reset to nominal conditions before a new simulation run. To prepare for the reinitialization of the simulation, a button on the CIM Unit is pressed, causing the time stamp value sent over the network to the TI Explorer to be reset to 0.0 and remain at that value as long as the button is depressed. When a time stamp value of 0.0 is first received by the GUI, the screens are all reset and all flags and internal variables reinitialized. The GUI loops, reading the buffer, mouse and keyboard input as usual but does not plot in the plotting window nor save past data values until the first nonzero time stamp is received. Thus, only the data which are considered valid are plotted. This feature is also implemented in the stand-alone mode since the two modes are identical except for the data acquisition portion. When running stand-alone, the user can set a logical flag from any output window which will set and hold the time-stamp value at 0.0 and, likewise, the user can reset the logical flag to restart the progression of time. A block diagram of this automatic reset procedure is shown in figure 12. Note that as long as the time-

stamp is 0.0, the previous data values are overwritten with the new set of variable values as the program loops. This way, once time starts advancing, the values plotted corresponding to a time-stamp of 0.0 are the ones received most recently.

SPEED

The SSME simulation is slowed down to about 10% of real time in order to accommodate some of the slower ICS hardware and software, most notably G2™. The CIM Unit sends data over the ethernet to the GUI at a rate of 20 packets per second which is sufficient to display the transient plots of the slowed down simulation. With this transfer rate, the curves which appear in the plotting window are displayed for nine seconds (180 points divided by 20 points per second equals 9 seconds) between each leftward shift. As the simulation is sped up to approach real time, a trade-off develops between the need to plot sufficient data points to show the response in detail, and the need to keep the display on the screen long enough to see it properly. Another consideration is that if the data transfer rate becomes too high, the Explorer might not be able to read all of the data packets and lose some when the input buffer overflows. This is especially likely to happen when a new screen is selected using the mouse since there is a lot of software overhead associated with exposing a different frame. Exposing a new screen might take a second or two during which time the CIM Unit continues to send data. Because of the linear interpolation feature of the plotting window, even if some data are lost, the remaining points are connected by straight lines which masks the fact that some are missing. During transients, however, this has the potential to be misleading because the sampling process filters out some high frequency information. Therefore, it is important to select a data transfer rate to the GUI which allows the important frequencies to be displayed and balance this against the ability of the Explorer to receive data packets without its buffer overflowing. If the data transfer frequency is low enough that the plot can be read easily, the transfer rate should be well within the range the Explorer can accept without a problem.

ADDING SCREENS

The GUI is modular. There are pieces of code corresponding to the creation of each frame. Copying the code and making minor changes such as to the names of the windows is all that is required to allow the construction of a screen. In order to be able to expose the screen, it is necessary to put an icon in the mouse-sensitive graphics window of an existing screen which, when clicked on, will bring up the new frame. Once the window is created, a picture containing mouse-sensitive icons can be drawn in the upper window, and axes can be inserted in the lower right window.

Using the existing code as a guide, it is straightforward to create additional screens. Thus the GUI can be extended as needed without a great deal of effort. However, creating the detail required for the picture in the mouse-sensitive graphics window might require a tremendous amount of work. It can take hundreds of graphical primitives to build up a figure such as that in the SSME screen (figure 4) and this does not even take into account the complexities involved with customizing the animation which might be desired as in the valve or sensor screens (figure 5 and 7, respectively).

CONCLUSIONS

A GUI has been developed and applied to the intelligent control system test bed of a reusable rocket engine. The GUI runs well in the testbed system. It is fast enough to meet the requirements of this demonstration which runs at 1/10th real-time. Each failure which the ICS is able to identify is displayed clearly on at least one screen. Additional screens can be added with little effort as they become necessary. Moving through the GUI is logical and straightforward because each graphical display is uncluttered yet detailed enough to show important information in an easy-to-understand format.

APPENDIX: FAILURES

BIT NUMBER	FAILURE
0	LPFTP Failure
1	HPFTP Tip Seals Failure
2	HPFTP Blade Failure
3	HPOTP Failure
4	FPOV Failure
5	MFV Failure
6	CCV Failure
7	MOV Failure
8	OPFV Failure
9	OPOV Failure
10	T_5 Sensor Failure
11	P_5 Sensor Failure
12	P_C Sensor Failure
13	P_{ft2d} Sensor Failure
14	S_{f2} Sensor Failure
15	T_{fd1} Sensor Failure
16	P_{fd1} Sensor Failure
17	Q_{fd1} Sensor Failure
18	S_{f1} Sensor Failure

REFERENCES

1. Rockwell International Corporation, Rocketdyne Division, SSME Orientation (Part A - Engine), Course No. ME-110(A)RIR, 1 December 1986.

2. Simon, D. L.; Wong, E.; Musgrave, J. L., "Implementation of an Intelligent Control System," Proceedings of the Advanced Earth-to-Orbit Propulsion Technology Conference, Huntsville, AL, May 19-21, 1992.

3. Rockwell International Corporation, "Engine Balance and Dynamic Model," Report FSCM No. 02602, Spec. No. RL00001, 1981.

4. Musgrave, J. L.; Paxson, D. E.; Litt, J. S.; Merrill, W. C., "A Demonstration of an Intelligent Control System for a Reusable Rocket Engine," Proceedings of the Advanced Earth-to-Orbit Propulsion Technology Conference, Huntsville, AL, May 19-21, 1992.

5. DeLaat, J. C.; Soeder, J. F., "Design of a Microprocessor-Based Control, Interface and Monitoring (CIM) Unit for Turbine Engine Controls Research," NASA TM-83433, 1983.

6. Merrill, W. C.; Musgrave, J. L.; Guo, T.-H., "Integrated Health Monitoring and Controls for Rocket Engines," NASA TM 105763, Proceedings of the 1992 SAE Aerospace Atlantic, Dayton, Ohio, April 7-10, 1992.

7. Duyar, A.; Guo, T.-H.; Merrill, W.; Musgrave, J., "Implementation of a Model Based Fault Detection Technique for Actuation Faults for the Space Shuttle Main Engine," Proceedings of the Third Annual Conference on Health Monitoring for Space Propulsion Systems, Cincinnati, Ohio, November 13-14, 1991, NASA TM 105781.

8. Gensym Corp., *G2 Reference Manual for G2 Version 2.0*," August, 1990.

9. Guo, T.-H., "An SSME High Pressure Oxidizer Turbopump Diagnostic System Using G2™ Real-Time Expert System," Proceedings of the Third Annual Conference on Health Monitoring for Space Propulsion Systems, Cincinnati, Ohio, November 13-14, 1991, NASA TM 105328.

10. Lin, C.-S.; Wu, I.-C.; Guo, T.-H., "Neural Networks for Sensor Failure Detection and Data Recovery," Proceedings of the International Conference on Artificial Neural Networks in Engineering, St. Louis, MO, November 10-12, 1991.

11. Litt, J.; Wong, E.; Simon, D. L., "A Prototype LISP-Based, Real-Time Object-Oriented Graphical User Interface for Control System Development," NASA TM , Month, year.

12. Schlaer, S.; Mellor, S. J., *Object-Oriented Systems Analysis: Modeling the World in Data*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.

13. Lorenzo, C. F.; Saus, J. R.; Ray, A.; Carpino, M.; Wu, M.-K., "Life Extending Control for Rocket Engines," NASA TM 105789, August 1992.

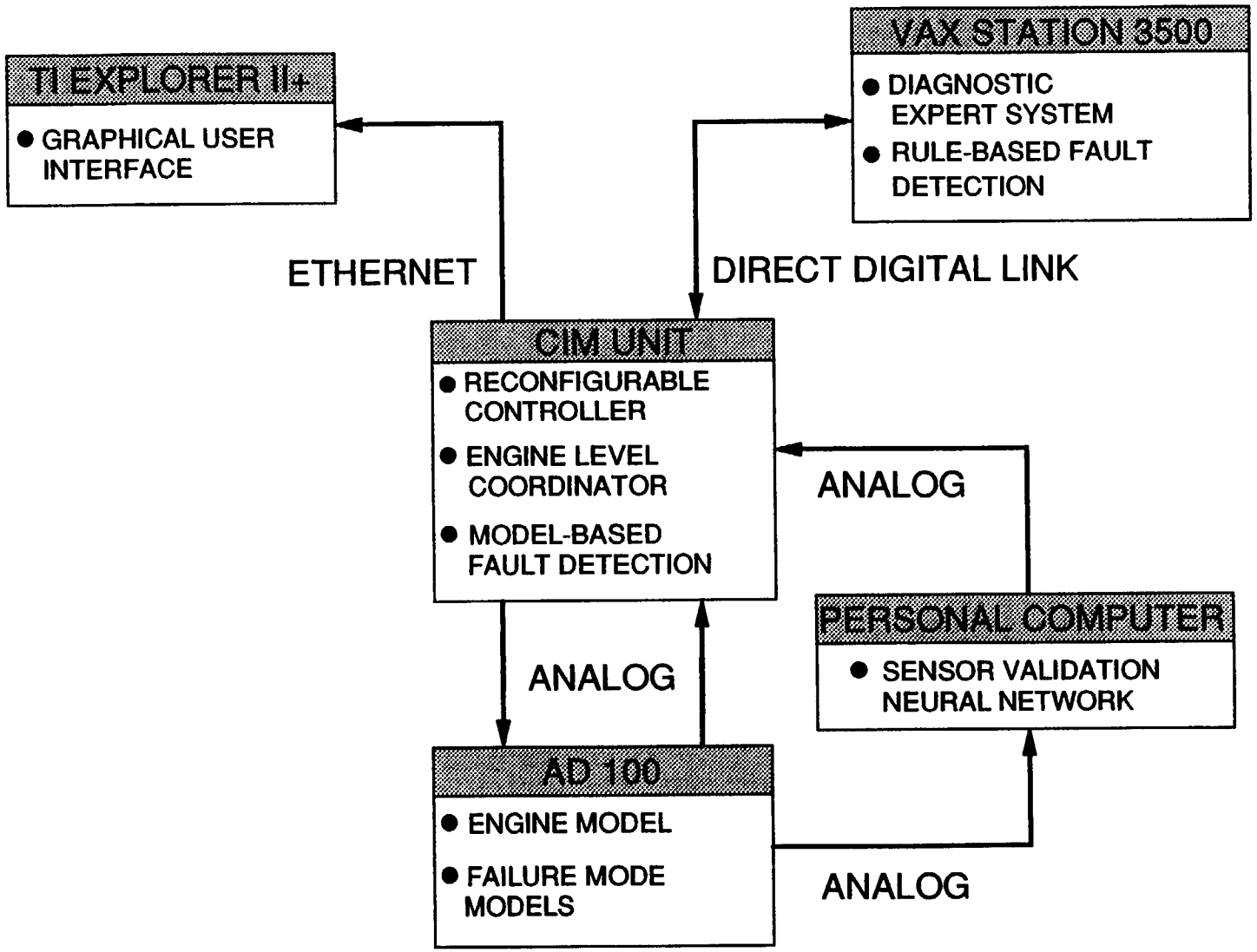

Figure 1.—Intelligent Control System simulation test bed.

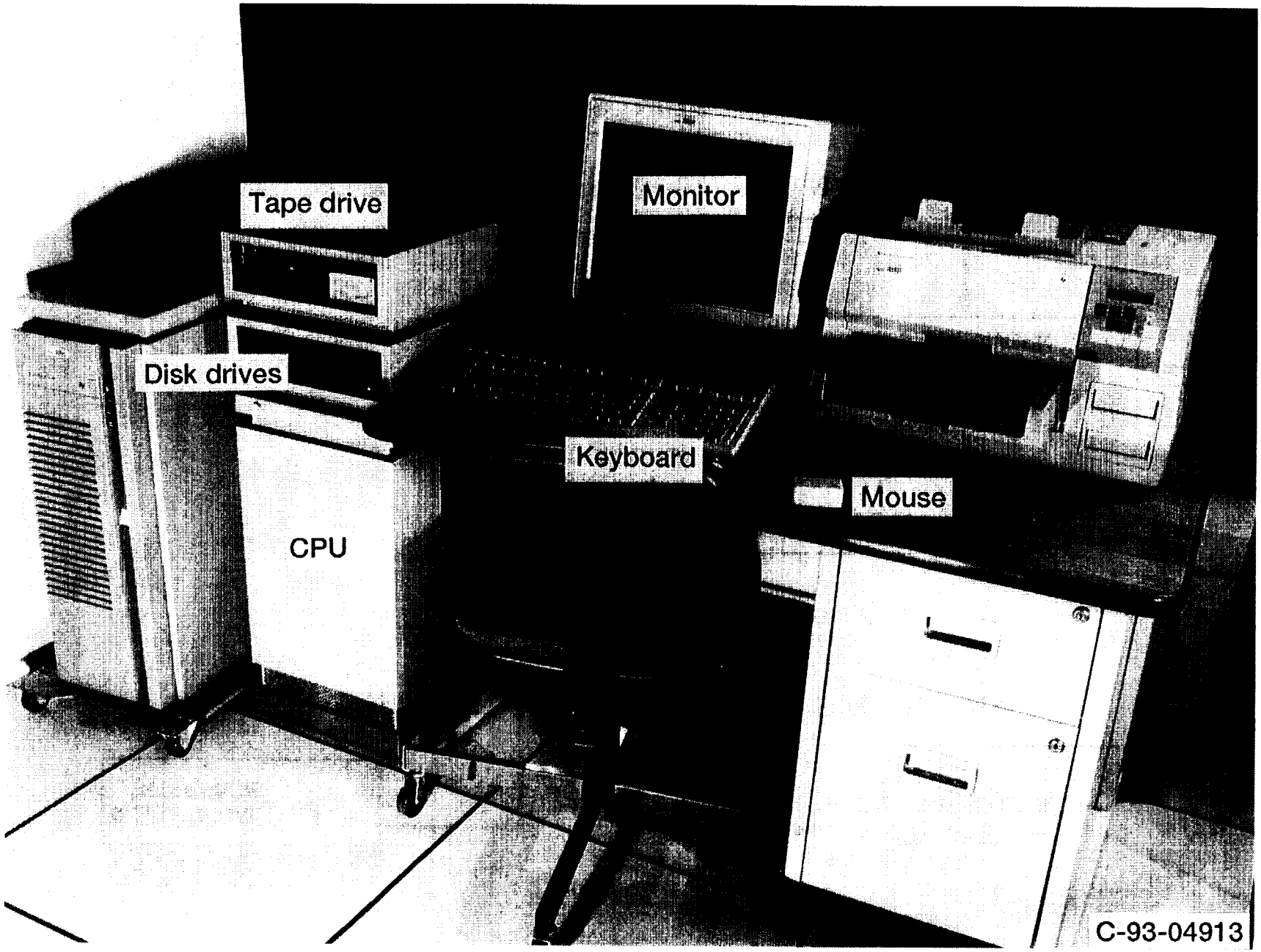

Figure 2.—Texas Instruments Explorer System.

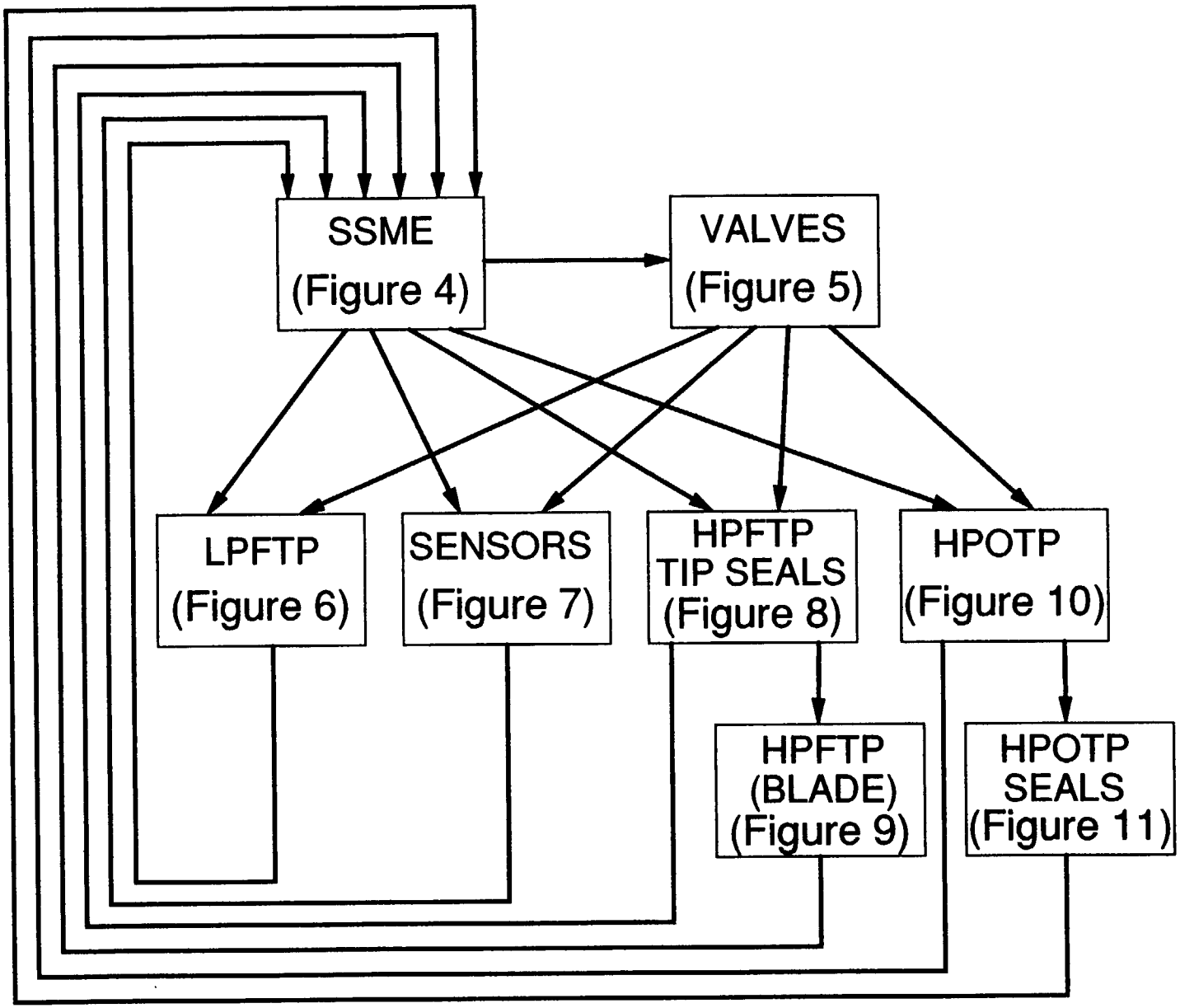

Figure 3.—Screen hierarchy for SSME ICS GUI.

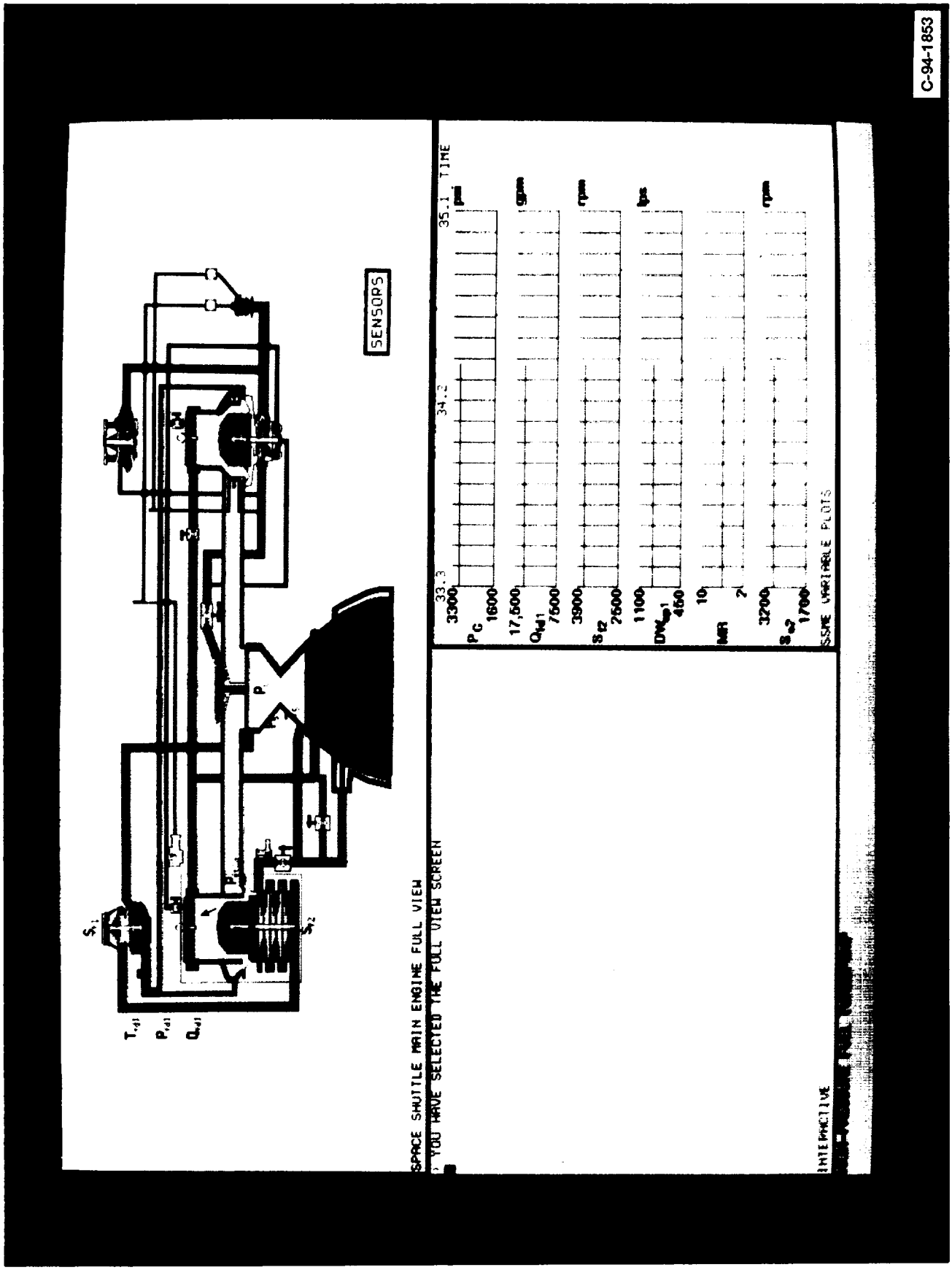

Figure 4.—SSME screen showing whole system.

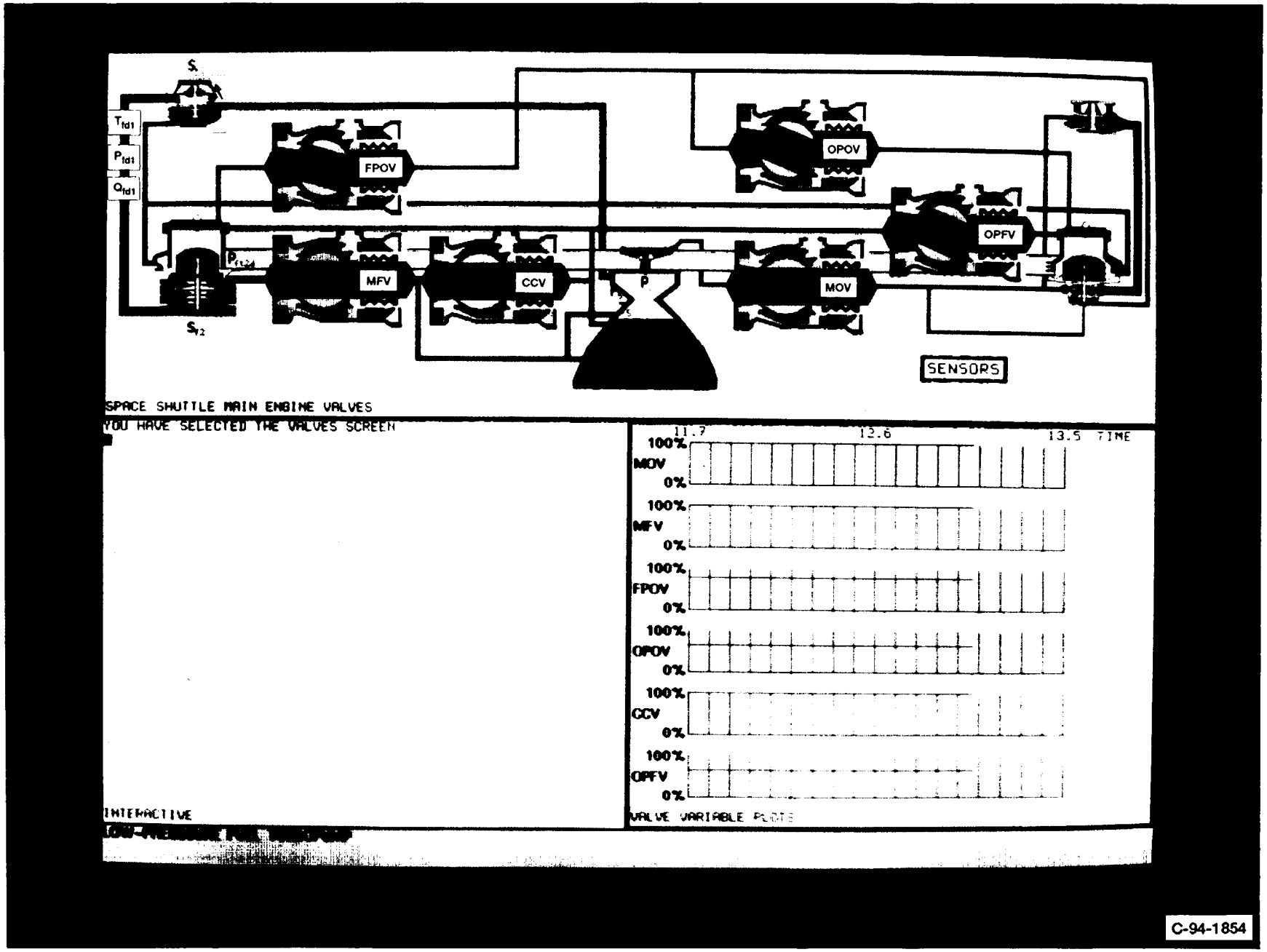

Figure 5.—Valve screen indicating percent open.

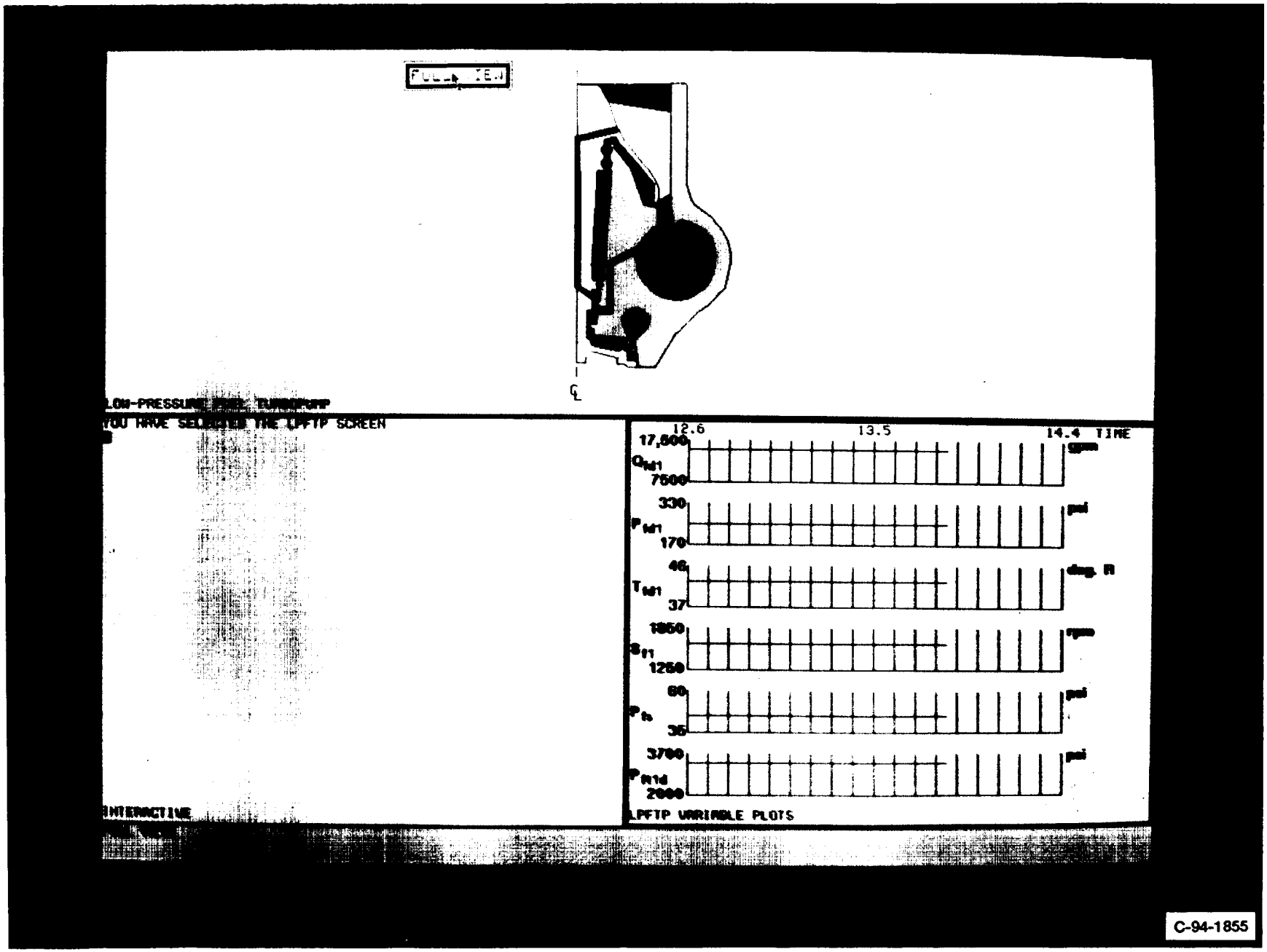

Figure 6.—LPFTP screen showing cross-section of pump.

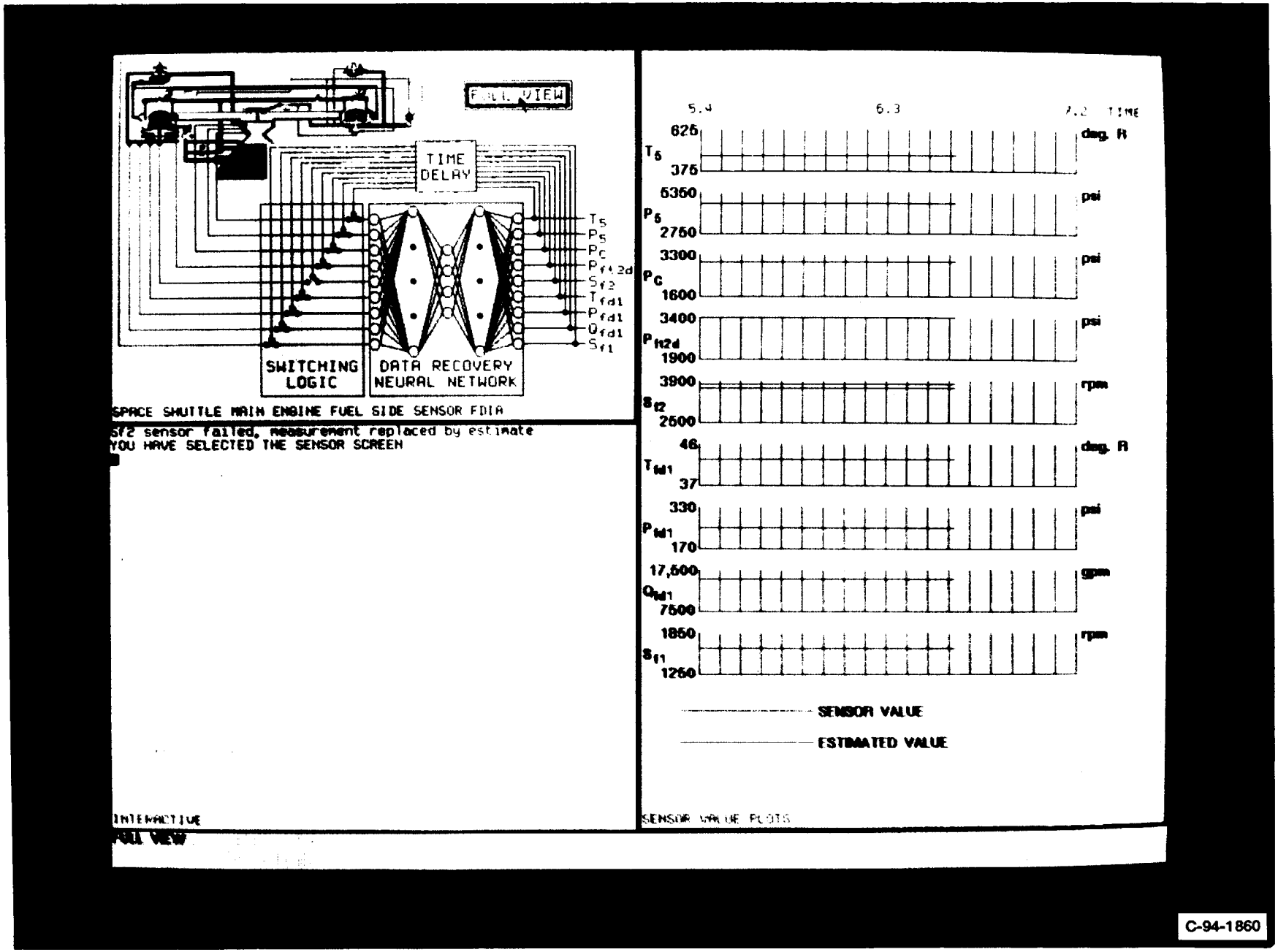

Figure 7.—Sensor screen symbolizing sensor FDIA.

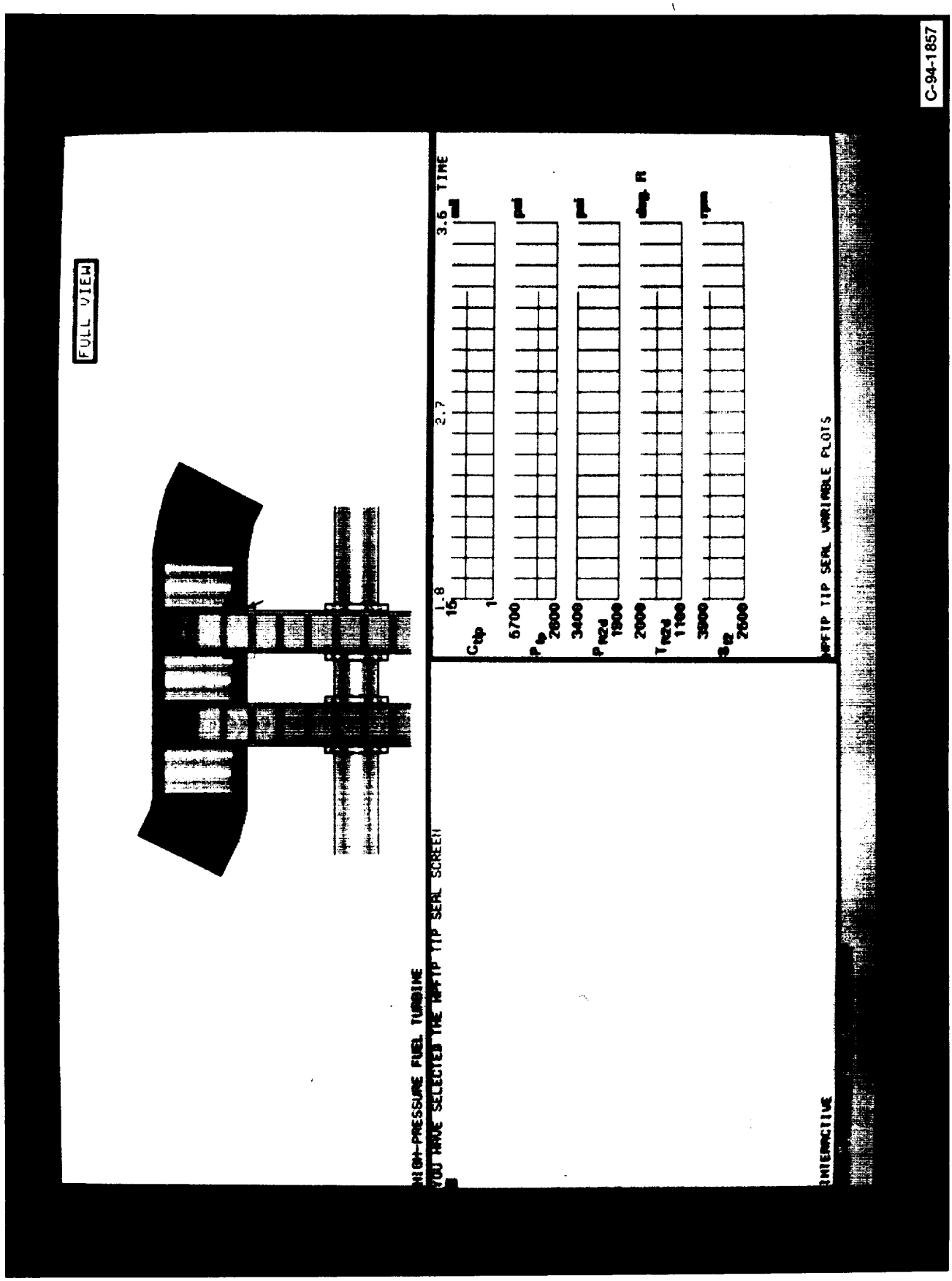

Figure 8.—HPFTP tip seal screen with rotating turbines.

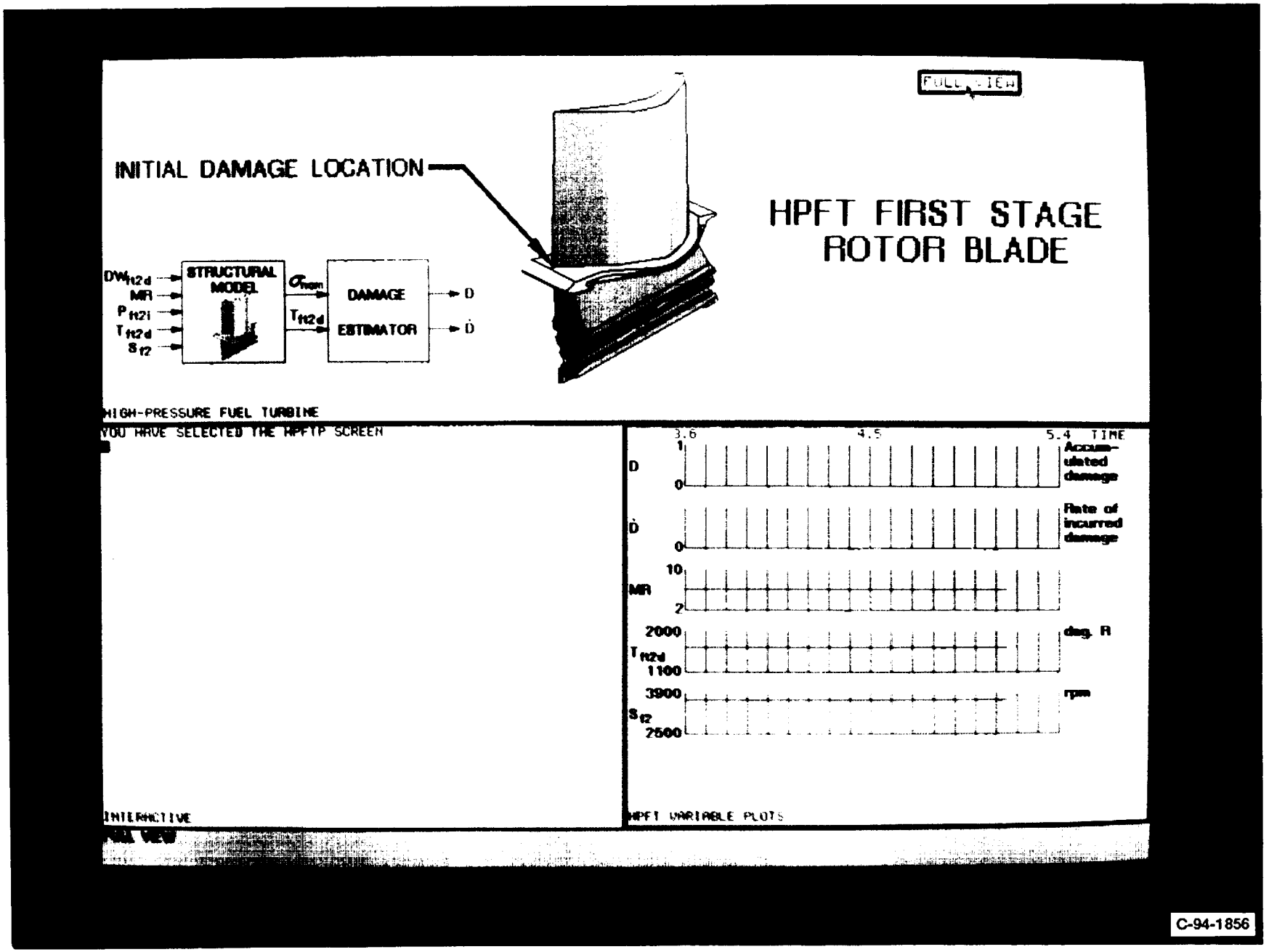

Figure 9.—HPFTP screen symbolizing remaining rotor blade life.

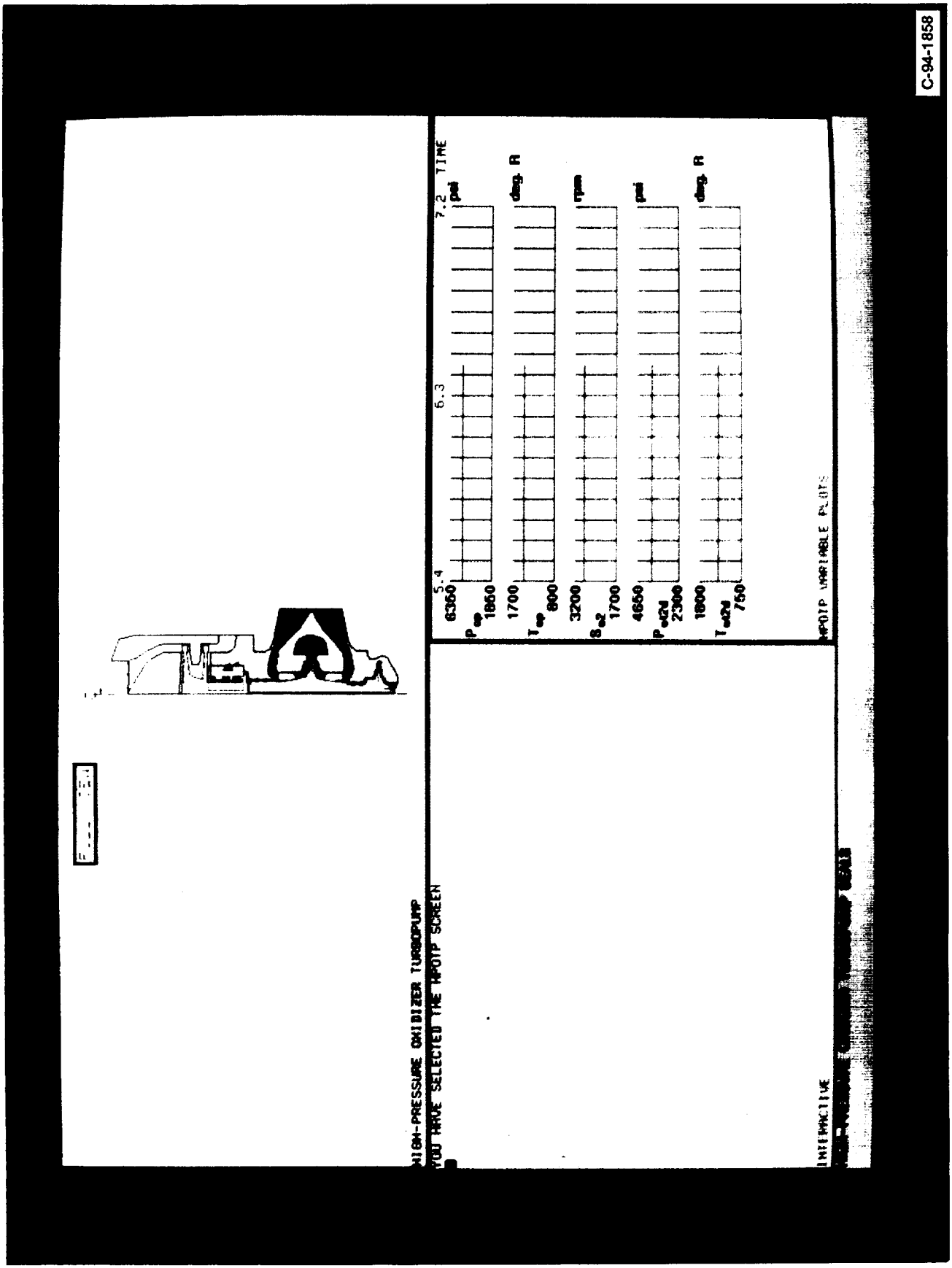

Figure 10.—HPOTP screen displaying location of seals.

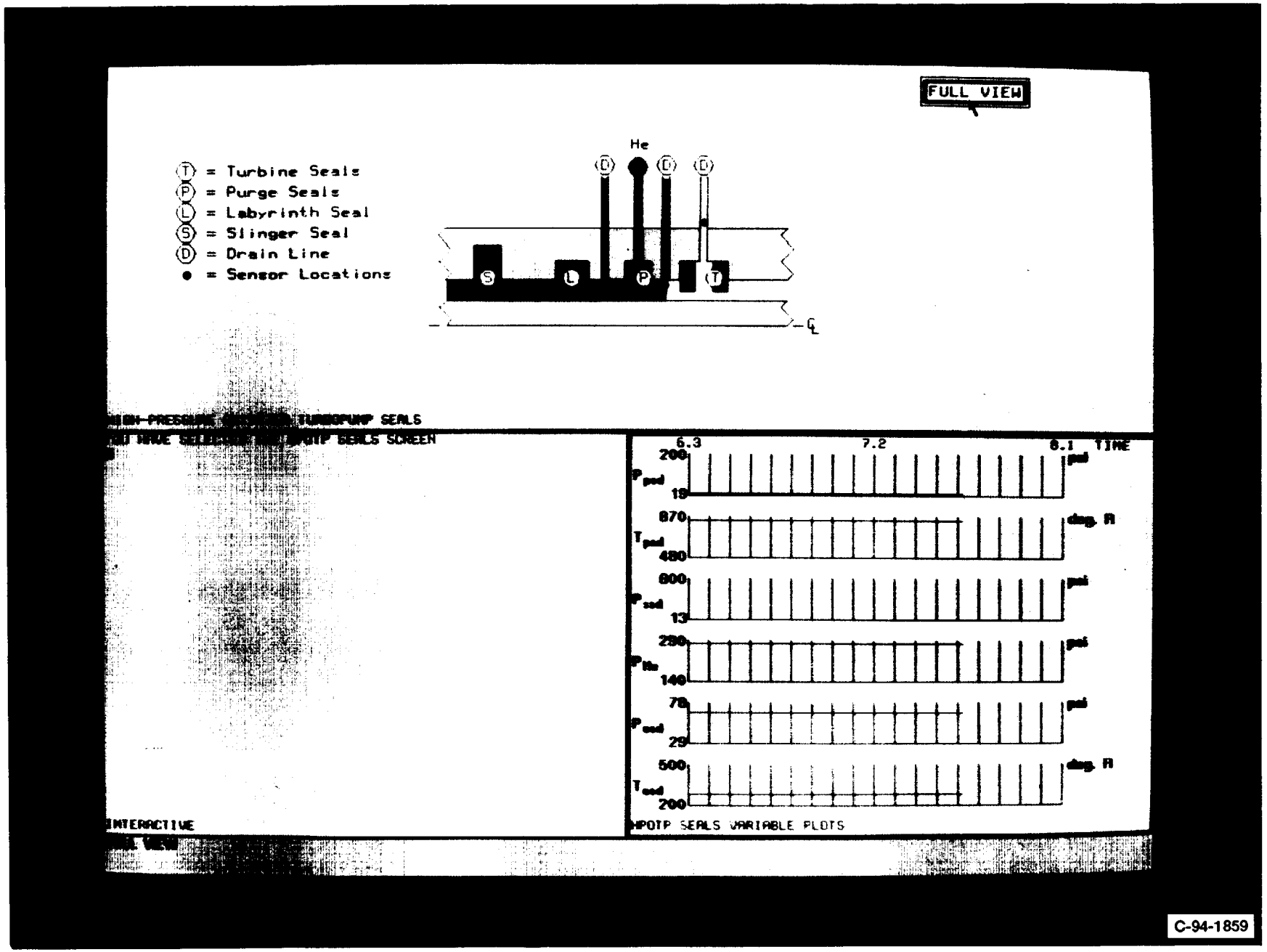

Figure 11.—HPOTP seals screen indicating leakage.

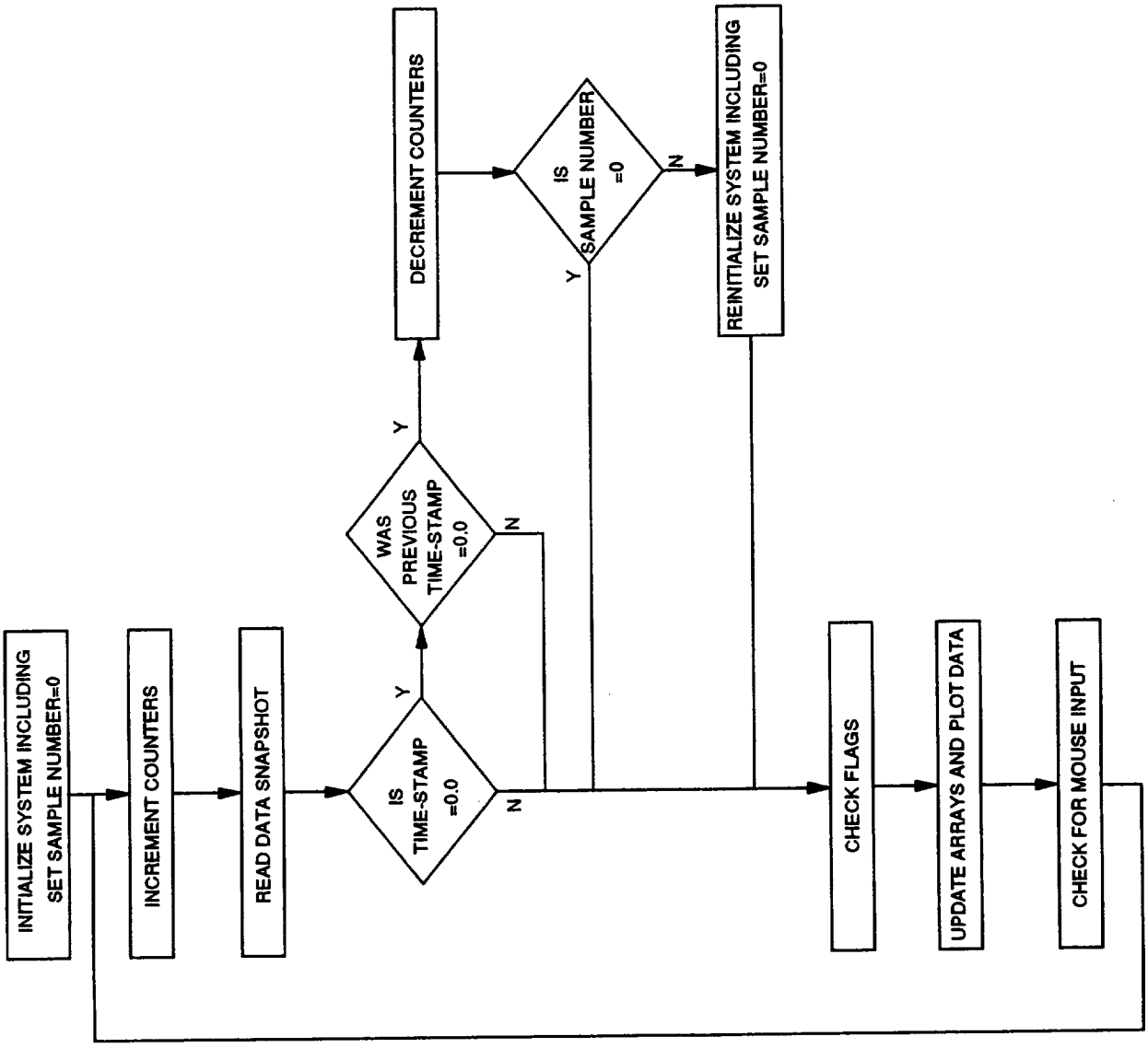

Figure 12.—Block diagram of automatic reset procedure.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1994	Technical Memorandum

4. TITLE AND SUBTITLE

An Object-Oriented Graphical User Interface for a Reusable Rocket Engine Intelligent Control System

5. FUNDING NUMBERS

WU–468–01–11
1L161102AH45

6. AUTHOR(S)

Jonathan S. Litt, Jeffrey L. Musgrave, Ten-Huei Guo, Daniel E. Paxson, Edmond Wong, Joseph R. Saus, and Walter C. Merrill

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Lewis Research Center
Cleveland, Ohio 44135–3191
and
Vehicle Propulsion Directorate
U.S. Army Research Laboratory
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–9166

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546–0001
and
U.S. Army Research Laboratory
Adelphi, Maryland 20783–1145

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM–106750
ARL–TR–598

11. SUPPLEMENTARY NOTES

Jonathan S. Litt, Vehicle Propulsion Directorate, U.S. Army Research Laboratory, NASA Lewis Research Center; Jeffrey L. Musgrave, Ten-Huei Guo, Daniel E. Paxson, Edmond Wong, Joseph R. Saus, and Walter C. Merrill, NASA Lewis Research Center. Responsible person, Jonathan S. Litt, organization code 0300, (216) 433–3748.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 20

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

An Intelligent Control System for Reusable Rocket Engines under development at NASA Lewis Research Center requires a graphical user interface to allow observation of the closed-loop system in operation. The simulation testbed consists of a real-time engine simulation computer, a controls computer, and several auxiliary computers for diagnostics and coordination. The system is set up so that the simulation computer could be replaced by the real engine and the change would be transparent to the control system. Because of the hard real-time requirement of the control computer, putting a graphical user interface on it was not an option. Thus, a separate computer used strictly for the graphical user interface was warranted. An object-oriented LISP-based graphical user interface has been developed on a Texas Instruments Explorer II+ to indicate the condition of the engine to the observer through plots, animation, interactive graphics, and text.

14. SUBJECT TERMS

Intelligent control; Reusable rocket engines; Graphical user interface

15. NUMBER OF PAGES

33

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration

Lewis Research Center
21000 Brookpark Rd.
Cleveland, OH 44135-3191